G-TAD: Sub-Graph Localization for Temporal Action Detection

https://www.deepgcns.org/app/g-tad

Mengmeng Xu, Chen Zhao, David S. Rojas, Ali Thabet, Bernard Ghanem
King Abdullah University of Science and Technology (KAUST), Saudi Arabia
{mengmeng.xu, chen.zhao, davidsantiago.blanco, ali.thabet, bernard.ghanem}@kaust.edu.sa

Abstract

Temporal action detection is a fundamental yet challenging task in video understanding. Video context is a critical cue to effectively detect actions, but current works mainly focus on temporal context, while neglecting semantic context as well as other important context properties. In this work, we propose a graph convolutional network (GCN) model to adaptively incorporate multi-level semantic context into video features and cast temporal action detection as a sub-graph localization problem. Specifically, we formulate video snippets as graph nodes, snippet-snippet correlations as edges, and actions associated with context as target sub-graphs. With graph convolution as the basic operation, we design a GCN block called GCNeXt, which learns the features of each node by aggregating its context and dynamically updates the edges in the graph. To localize each sub-graph, we also design an SGAlign layer to embed each sub-graph into the Euclidean space. Extensive experiments show that G-TAD is capable of finding effective video context without extra supervision and achieves state-of-the-art performance on two detection benchmarks. On ActivityNet-1.3, it obtains an average mAP of 34.09%*; on THUMOS14, it reaches* 51.6% *at IoU@0.5 when combined with a proposal processing method. G-TAD code is publicly available at https://github.com/frostinassiky/gtad.*

1. Introduction

Video understanding has gained much attention from both academia and industry over recent years, given the rapid growth of videos published in online platforms. Temporal action detection is one of the interesting but challenging tasks in this area. It involves detecting the start and end frames of action instances, as well as predicting their class labels. This is onerous especially in long untrimmed videos.

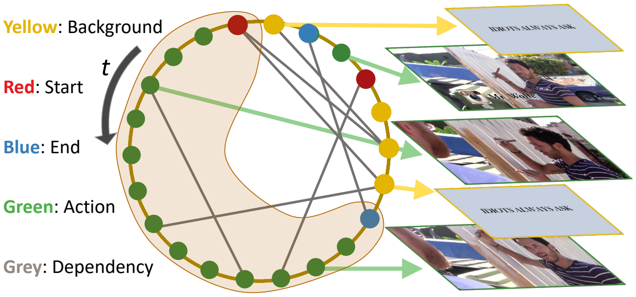

Figure 1. **Graph formulation of a video.** Nodes: video snippets (a video snippet is defined as consecutive frames within a short time period). Edges: snippet-snippet correlations. Sub-graphs: actions associated with context. There are 4 types of nodes: action, start, end, and background, shown as colored dots. There are 2 types of edges: (1) temporal edges, which are pre-defined according to the snippets' temporal order; (2) semantic edges, which are learned from node features.

Video context is an important cue to effectively detect actions. Here, we refer to context as frames that do not belong to the target action but carry its valuable indicative information. Using video context to infer potential actions is natural for human beings. In fact, empirical evidence shows that humans can reliably guess or predict the occurrence of a certain type of action by only looking at short video snippets where the action does not happen [1, 2]. Therefore, incorporating context into temporal action detection has become an important strategy to boost detection accuracy in the recent literature [11, 15, 9, 33, 44, 56, 30]. Researchers have proposed various ways to take advantage of video context, such as extending temporal action boundaries by a predefined ratio [11, 15, 44, 56, 30], using dilated convolution to encode context into features [9], and aggregating context features implicitly by way of a Gaussian curve [33]. All these methods only utilize *temporal context*, which precedes or follows an action instance in its immediate temporal neighborhood. However, real-world videos vary dramatically in temporal extent, action content, and even editing preferences. *Temporal context* does not fully exploit the rich merits of video context, and it may even impair detection accuracy if not properly designed for underlying videos.

So, what properties characterize desirable video context for the purpose of accurate action detection? First, context should be semantically correlated to the target action other than merely temporally located in its vicinity. Imagine the case where we manually stitch an action clip into some irrelevant frames, the abrupt scene change surrounding the action would definitely not benefit the action detection. On the other hand, snippets located at a distance from an action but containing similar semantic content might provide indicative hints for detecting the action. Second, context should be content-adaptive rather than manually pre-defined. Considering the vast variation of videos, context that helps to detect different action instances could be different in lengths and locations based on the video content. Third, context should be based on multiple semantic levels, since using only one form/level of context is unlikely to generalize well.

We endow video context with all the above properties by casting action detection as a sub-graph localization problem based on a graph convolutional network (GCN) [24]. We represent each video sequence as a graph, each snippet as a node, each snippet-snippet correlation as an edge, and target action associated with context as sub-graph, as shown in Fig. 1. The context of a snippet is considered to be all snippets connected to it by edges in a video graph. We define two types of edges — temporal edges and semantic edges, corresponding to temporal context and semantic context, respectively. Temporal edges exist between each pair of adjacent snippets, whereas semantic edges are dynamically learned from the video features at each layer. Hence, multi-level context of each snippet is gradually aggregated into the features of the snippet throughout the entire GCN.

The pipeline of our proposed Graph-Temporal Action Detection method, dubbed G-TAD, is analogous to faster R-CNN [17, 35] in object detection. There are two critical designs in G-TAD. First, the GCN-based feature extraction block GCNext, which is inspired by ResNeXt [49], generates context-enriched features. It corresponds to the CNN blocks of the backbone network in faster R-CNN. Second, to mimic region of interest (RoI) alignment [19], we design a sub-graph of interest alignment layer SGAlign to generate a fixed-size representation for each sub-graph and embed all sub-graphs into the same Euclidean space. Finally, we apply a classifier on the features of each sub-graph to obtain detection. We summarize our contributions as follows.

(1) We present a novel GCN-based video model to fully exploit video context for effective temporal action detection. Using this video GCN representation, we are able to adaptively incorporate multi-level semantic context into the features of each snippet.

(2) We propose G-TAD, a new sub-graph detection framework to localize actions in video graphs. G-TAD includes two main modules: GCNeXt and SGAlign. GCNeXt performs graph convolutions on video graphs, leveraging both temporal and semantic context. SGAlign re-arranges sub-graph features in an embedded space suitable for detection.

(3) G-TAD achieves state-of-the-art performance on two popular action detection benchmarks. On ActivityNet-1.3, it achieves an average mAP of 34.09%. On THUMOS14 it reaches 51.6% at IoU@0.5 when combined with a proposal processing method.

2. Related Work

2.1. Video Representation

Action Recognition. Many CNN based methods have been proposed to address the action recognition task. Two-stream networks [14, 38, 43] use 2D CNNs to extract frame features from RGB and optical flow sequences. These 2D CNNs can be designed from scratch [20, 39] or pre-trained on image recognition tasks [12]. Other methods [41, 8, 34, 52] use 3D CNNs to encode spatio-temporal information from the original video. In our work, we use the pre-trained action recognition model in [51, 45] to extract video snippet features as G-TAD input.

Action Detection. Temporal action detection is to predict the boundaries and categories of action instances in untrimmed videos. A common practice is to first generate temporal proposals and then classify each proposal into one of the action categories [37, 40, 56, 55, 9, 30]. For proposal generation, they either use fixed handcrafted anchors [5, 6, 13, 15, 37] , or adaptively form proposal candidates by connecting potential start and end frames [56, 30]. G-TAD uses anchors to define sub-graphs, but also incorporates start/end prediction to regularize the training process.

2.2. GCN in Videos

Graphs in Video Understanding. Graphs have been widely used for data/feature representation in various video understanding tasks, such as action recognition [31, 47, 10], and action localization [55]. In action recognition, Liu *et al.* [31] view a video as a 3D point cloud in the spatial-temporal space. Wang *et al.* [47] represent a video as a space-time region graph, in which the graph nodes are defined by object region proposals. In action detection, Zeng *et al.* [55] consider temporal action proposals as nodes in a graph, and refine their boundaries and classification scores based on the established proposal-proposal dependencies. Differently from previous works, G-TAD takes video snippets as nodes in a graph and form edges between them based on both their temporal ordering and semantic similarity.

Graph Convolutions Networks. Graph Convolutional Networks (GCNs) [24] are widely used for non-Euclidean structures. In these years, its successful application has been seen in computer vision tasks due to their versatility and effectiveness, such as 3D object detection [18] and point cloud segmentation [48, 50]. Meanwhile, various GCN architectures are proposed for more effective and flexible modelling. One representative work is the edge convolution method by Wang *et al.* [48] for point clouds. It com-

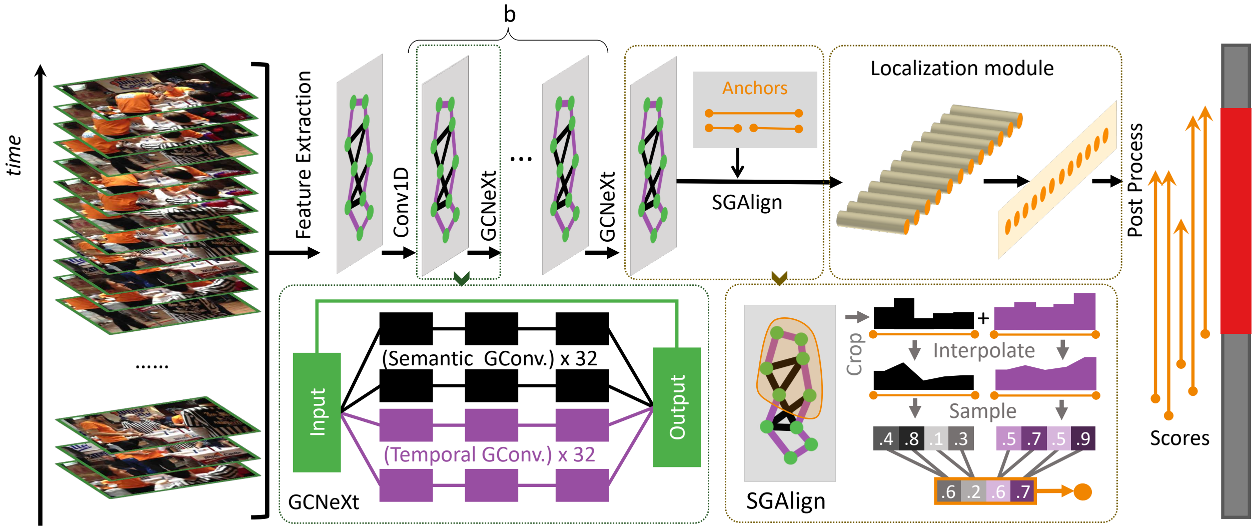

Figure 2. **Overview of G-TAD architecture.** The input is a sequence of snippet features. We first extract features using b GCNeXt blocks, which gradually aggregate both temporal and multi-level semantic context. Semantic context, encoded in semantic edges, is dynamically learned from features at each GCNeXt layer. Then we feed the extracted features into the SGAlign layer, where sub-graphs defined by a set of anchors are represented by fixed-size features. Finally, the localization module scores and ranks the sub-graphs for detection.

putes graph edges (represented as node adjacency) at each graph layer based on the feature distance between nodes, and enriches the node feature by aggregating the features over the neighbourhood as node output. Recently, Li *et al.* [25, 26] propose DeepGCNs to enable GCNs to go as deep as 100 layers using residual/dense graph connections and dilated graph convolutions, and explore ways to automatically design GCNs [27]. G-TAD uses a DeepGCN-like structure to apply graph convolutions on a dynamic semantic graph as well as a fixed temporal graph.

3. Proposed Method

3.1. Problem Formulation

The input to our pipeline is a video sequence of l_v frames. Following recent video action proposal generation methods [5, 13, 15, 30], we construct our G-TAD model using feature sequences extracted from raw video frames. We average the features of every σ consecutive frames and refer to each set of the σ frames as a **snippet**. In this way, our input visual feature sequence is represented by $X^{(0)} \in \mathbb{R}^{C \times L}$, where C is the feature dimension of each snippet, and L is the number of snippets. Each video sequence has a set of N annotations $\Psi = \{\psi_n = (t_{s,n}, t_{e,n}, c_n)\}_{n=1}^{N}$, where ψ_n represents an action instance, and $t_{s,n}$, $t_{e,n}$, and c_n are its start time, end time, and action class, respectively.

The temporal action detection task is to predict M possible actions $\Phi = \left\{\phi_m = \left(\hat{t}_{s,m}, \hat{t}_{e,m}, \hat{c}_m, p_m\right)\right\}_{m=1}^{M}$ from V. Here, $(\hat{t}_{s,m}, \hat{t}_{e,m})$ represents the predicted temporal boundaries for the m^{th} predicted action; $\hat{c}_m$ and p_m are its predicted action class and confidence score, respectively.

3.2. G-TAD Architecture

Our action detection framework is illustrated in Fig. 2. We feed the snippet features $X^{(0)}$, into a stack of b GCNeXt blocks, which is designed inspired by ResNeXt [49] to obtain context-aware features. Each GCNeXt contains two graph convolution streams. One stream operates on fixed temporal neighbors, and the other adaptively aggregates semantic context into snippet features. Both streams follow the split-transform-merge strategy with multiple convolution paths [49] (the number of paths is defined as cardinality) to generate updated graphs, which are aggregated into one graph as the block output. At the end of all b GCNeXt blocks, we extract a set of sub-graphs based on the pre-defined temporal anchors (see Section 4.2).

Then we have the sub-graph of interest alignment layer SGAlign to represent each sub-graph using a feature vector. In the end, we use multiple fully connected layers to predict the intersection over union (IoU) of the feature vector representing every sub-graph and the ground truth action instance. We provide a detailed description of both GCNeXt and SGAlign in Sections 3.3 and 3.4, respectively.

3.3. GCNeXt for Context Feature Encoding

Our basic graph convolution block, GCNeXt, operates on a graph representation of the video sequence. It encodes snippets using their temporal and semantic neighbors. Fig. 3 illustrates the architecture of GCNeXt.

We build a video graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_l\}_{l=0}^{L}$

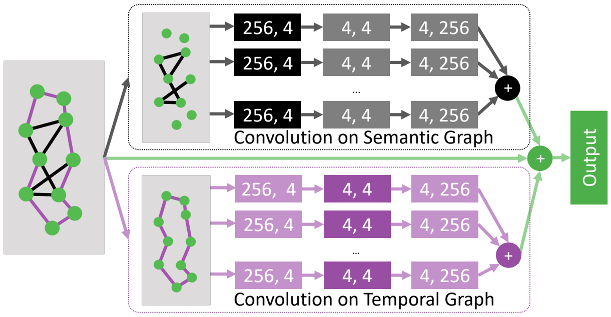

Figure 3. **GCNeXt block.** The input feature is processed by temporal and semantic streams with the same cardinality. Black and purple boxes represent operations in the temporal and semantic streams, respectively, darker colors referring to graph convolutions and lighter ones 1-by-1 convolutions. The numbers in each box refer to input and output channels. Both streams follow a split-transform-merge strategy with 32 paths each to increase the diversity of transformations. The module output is the summation of both streams and the input.

and $\mathcal{E} = \mathcal{E}_t \cup \mathcal{E}_s$ denote the node and edge sets, respectively. In this case, each node represents a snippet and each edge shows a dependency between a pair of snippets. We define two types of edges — temporal edges $\mathcal{E}_t$ and semantic edges $\mathcal{E}_s$. Accordingly we have the temporal stream and the semantic stream. We describe each type of edge as well as the graph convolution process in the following.

Temporal Edges ($\mathcal{E}_t$) encode the temporal order of video snippets. Each node $v_i \in \mathcal{V}$ has one unique forward edge to node v_{i+1}, and one backward edge to node v_{i-1}. In this case, we have $\mathcal{E}_t = \mathcal{E}_t^f \cup \mathcal{E}_t^b$, where $\mathcal{E}_t^f$ and $\mathcal{E}_t^b$ are forward and backward temporal edge sets defined as follows:

$$\mathcal{E}_t^f = \{(v_i, v_{i+1}) |\, i \in \{1, 2, \dots, L-1\}\}, \qquad (1)$$

$$\mathcal{E}_t^b = \{(v_i, v_{i-1}) |\, i \in \{2, \dots, L-1, L\}\}, \qquad (2)$$

where L is the number of snippets in the video.

Semantic Edges ($\mathcal{E}_s$) are defined from the notion of dynamic edge convolutions [48], which dynamically constructs edges between graph nodes according to their feature distances. The goal of our semantic edges is to collect information from semantically correlated snippets. we define the semantic edge set $\mathcal{E}_s$ for each node v_i in $\mathcal{G}$ as follows

$$\mathcal{E}_s = \{(v_i, v_{n_i(k)}) | i \in \{1, 2, \dots, L\}; k \in \{1, 2, \dots K\}\}.$$

Here, $n_i(k)$ refers to the node index of the k^{th} nearest neighbor of node v_i. It is determined dynamically at every GCNeXt block in the node feature space, enabling us to update the nodes that intrinsically carry semantic context information throughout the network. Therefore, $\mathcal{E}_s$ adaptively changes to represent new levels of semantic context.

Graph Convolution. We use $X = [x_1, x_2, \dots, x_L] \in \mathbb{R}^{C \times L}$ to represent the features for all the nodes in the graph and transform it using the graph convolution operation $\mathcal{F}$. There are several choices for $\mathcal{F}$ in the literature. For simplicity, we use a single-layer edge convolution [48] as our graph convolution operation:

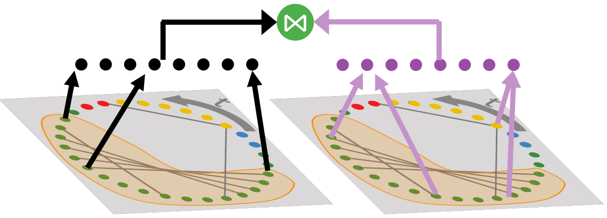

Figure 4. **SGAlign layer**. SGAlign extracts sub-graph features based on both GCNeXt features (left) and semantic features (right), and concatenates both sub-graph features as output. The dots on top represent sub-graph features. On the bottom, the dots represent graph nodes, grey lines are semantic edges, and the orange highlighted zones are sub-graphs. Note that since the semantic feature of each node is computed using its neighbors, each entry in the the sub-graph feature essentially corresponds to multiple semantically correlated nodes in the graph.

$$\mathcal{F}(X, A, W) = ([X^T, AX^T - X^T]W)^T. \qquad (3)$$

Here, $W \in \mathbb{R}^{2C \times C'}$ is trainable weight; $A \in \mathbb{R}^{L \times L}$ is the adjacency matrix without self-loops (*i.e.* edges between a node and itself); $[\cdot, \cdot]$ represents the matrix concatenation of columns. We formulate the $(i, j)^{\text{th}}$ element in A as $A_{(i,j)} = \mathbf{1}\{(v_i, v_j) \in \mathcal{E}\}$, where $\mathbf{1}\{\cdot\}$ is the indicator function.

Either stream in GCNeXt leverages a split-transform-merge strategy [49] with 32 paths to increase the diversity of transformations. Each path contains one graph convolution as in Eq. 3 and two 1-by-1 convolutions, their composition denoted as $\mathcal{F}'$.

Stream Aggregation. The GCNeXt output is the aggregation of semantic and temporal steams as well as the input, which can be formulated as:

$$\begin{aligned}\mathcal{H}(X, A, W) = ReLU(&\mathcal{F}'(X, A_t^f, W_t^f) + \mathcal{F}'(X, A_t^b, W_t^b) \\ &+ \mathcal{F}'(X, A_s, W_s) + X), \qquad (4)\end{aligned}$$

where A_t^f, A_t^b, and A_s are adjacency matrices, $W = \{W_t^f, W_t^b, W_s\}$ are the trainable weights, corresponding to $\mathcal{E}_t^f$, $\mathcal{E}_t^b$, and $\mathcal{E}_s$, respectively. $ReLU$ is the rectified linear unit as the activation function. In the **supplementary material**, we simplify Eq. 4 and prove that it can be efficiently computed by zero-padded 1D convolutions.

3.4. Sub-Graph Alignment and Localization

Sub-Graph of Interest Alignment (SGAlign). The GCNeXt blocks generate the features of all snippets $\{x_l\}_{l=1}^L$ (dubbed as GCNeXt features), which contains aggregated information from their temporal and semantic context. Using $\{x_l\}_{l=1}^L$, we obtain an updated graph $\{\mathcal{V}, \mathcal{E}\}$. In

Algorithm 1 Interpolation and Rescaling in SGAlign

```
Input: Features of all nodes in the entire graph $\{x_l\}_{l=1}^{L}$;
    sub-graphs $\{\mathcal{G}_{a_j}\}_{j=1}^{J}$, where $J$ is the total number of
    sub-graphs, $a_j = (t_{s,j}, t_{e,j})$; alignment quantity $\tau$;
for each sub-graph $\mathcal{G}_{a_j}$ do
    Arrange all nodes in $\mathcal{G}_{a_j}$ in their temporal order;
    Compute sub-graph size $d = t_{s,j} - t_{e,j}$, sampling
        interval $s = \lceil d/\tau \rceil$, interpolation quantity $T = \tau s$;
    Sample $T$ points based on linear interpolation using
        the two neighbors of each point $l$ = [$t_s + kd/T$ for $k$
        in range($T$)]
    $X_{\text{in}}$ = [$(\lceil i \rceil - i)x_{\lfloor i \rfloor} + (i - \lfloor i \rfloor)x_{\lceil i \rceil}$ for $i$ in $l$]
    $z_{a_j}$ = [mean($X_{in}[ks{:}(k+1)s]$) for $k$ in range($\tau$)]
end for
Output: $Z = \{z_{a_j}\}_{j=1}^{J}$.
```

SGAlign, we further exploit semantic context by averaging the neighbor features of each node, formulated as $y_l = \frac{1}{K}\sum_{k=1}^{K} x_{n_l(k)}$, and dub y_l as semantic features.

SGAlign uses pre-defined anchors to extract sub-graphs from $\{\mathcal{V}, \mathcal{E}\}$. Given each action anchor $a = (t_s, t_e)$, a sub-graph $\mathcal{G}_a$ is defined as a subset of $\mathcal{G}$ such that $\mathcal{G}_a = \{\mathcal{V}_a, \mathcal{E}_a\}$, where $\mathcal{V}_a = \{v_l \in \mathcal{V}\} | t_s \leq l \leq t_e\}$ and $\mathcal{E}_a = \{(v_i, v_j) \in \mathcal{E}_s | v_i \in \mathcal{V}_a\}$. For the sub-graph $\mathcal{G}_a$, we sample τ points (τ: alignment quantity) via interpolation and rescaling as described in Alg. 1, and generate the sub-graph feature $y_a \in \mathbb{R}^{\tau C}$, where C is the feature dimension.

We run Alg. 1 independently using the GCNeXt features $\{x_l\}_{l=1}^{L}$ and the semantic features $\{y_l\}_{l=1}^{L}$ as input. For the former, we sample τ_1 points and obtain the sub-graph features $z_{1a} \in \mathbb{R}^{\tau_1 C}$; and for the latter, we sample τ_2 points and obtain $z_{2a} \in \mathbb{R}^{\tau_2 C}$, respectively. We concatenate z_{1a} and z_{2a} as the output of the SGAlign layer. Fig. 4 illustrates the idea of SGAlign using the two features.

By explicitly using the semantic feature y_l, SGAlign adaptively aggregates semantic context information when computing the features of each sub-graph. This is essentially different from the methods that manually extend the anchor boundaries for incorporating temporal context [30, 56] and leads to superior performance.

It is worth mentioning that the sampling interval s is based on the sub-graph size d and alignment quantity τ, to ensure that the output z_{a_j} is the weighted sum of *all the nodes* in the sub-graph. In Sec. 4.4, we show that this sampling strategy gives us experimental improvements.

Sub-Graph Localization. For each sub-graph $\mathcal{G}_a$, we calculate its Intersection-over-Union (IoU) with all ground-truth actions ψ in Ψ, and denote the maximum IoU g_c as the training target. We apply three fully connected (FC) layers on top of the SGAlign layer for each sub-graph feature. The last FC layer has two output scores p_{cls} and p_{reg}, which are trained to match g_c using classification and regression losses, respectively.

3.5. G-TAD Training

We train G-TAD with the sub-graph localization loss L_g and the node classification loss L_n, as well as an $\mathcal{L}_2$-norm regularization loss L_r for all trainable parameters Θ:

$$L = L_g + L_n + \lambda_2 \cdot L_r, \tag{5}$$

where we set $\lambda_2 = 10^{-4}$. The loss L_g is used to determine the confidence scores of sub-graphs, and the loss L_n, classifying each node based on its location relative to an action, can drastically improve the network convergence.

Sub-Graph Localization Loss. The sub-graph loss L_g is defined as follows:

$$L_g = L_{wce}(p_{cls}, \mathbf{1}\{g_c > 0.5\}) + \lambda_1 \cdot L_{mse}(p_{reg}, g_c), \tag{6}$$

where L_{mse} is the mean square error loss and L_{wce} is the weighted cross entropy loss. The weight is computed to balance the positive and negative training samples. In our experiments, we take the trade-off coefficient $\lambda_1 = 10$, since the second loss term tends to be smaller than the first.

Node Classification Loss. Along with the sub-graph localization loss L_g, we use the loss L_n to classify each node in the whole graph based on whether they are start or end points of an action. We add FC layers after the first GCNeXt block to produce the start/end probabilities (p_s, p_e) (these layers are ignored at test time). We use (g_{ns}, g_{ne}) to denote the corresponding training targets for each node. We use the weighted cross entropy loss to compute the discrepancy between the predictions and the targets, and hence have L_n formulated as $L_n = L_{wce}(p_s, g_{ns}) + L_{wce}(p_e, g_{ne})$.

3.6. G-TAD Inference

At inference time, G-TAD predicts classification and regression scores for each sub-graph $\mathcal{G}_a$. From the J sub-graphs, we construct predicted actions $\Phi = \{\phi_j = (\hat{t}_{s,j}, \hat{t}_{e,j}, \hat{c}_j, p_j)\}_{j=1}^{J}$, where $(\hat{t}_{s,j}, \hat{t}_{e,j})$ refer to the predicted action boundaries, $\hat{c}_j$ is the predicted action class, and p_j is the fused confidence score of this prediction, computed as $p_j = p_{cls}^{\alpha} \cdot p_{reg}^{1-\alpha}$. In our experiments, we search for the optimal α in each setup.

4. Experiment

4.1. Datasets and Metrics

ActivityNet-1.3 [7] is a large-scale action understanding dataset for action recognition, temporal detection, proposal generation and dense captioning tasks. It contains 19,994 temporally annotated untrimmed videos with 200 action categories, which are divided into training, validation and testing sets by the ratio of 2:1:1.

THUMOS-14 [23] dataset contains 413 temporally annotated untrimmed videos with 20 action categories. We use the 200 videos in the validation set for training and evaluate on the 213 videos in the testing set.

Detection Metric. We take mean Average Precision (mAP) at certain IoU thresholds as the main evaluation metric. Following the official evaluation API, the IoU thresholds $\{0.5, 0.75, 0.95\}$ and $\{0.3, 0.4, 0.5, 0.6, 0.7\}$ are used for ActivityNet-1.3 and THUMOS-14, respectively. On ActivityNet-1.3, we also report average mAP over 10 different IoU thresholds $[0.5 : 0.05 : 0.95]$.

4.2. Implementation Details

Features and Anchors. We use pre-extracted features for both datasets. For ActivityNet-1.3, we adopt the pre-trained two-stream network by Xiong et. al. [51], with downsampling ratio $\sigma = 16$. Each video feature sequence is rescaled to $L = 100$ snippets using linear interpolation. For THUMOS-14, the video features are extracted using TSN model [44] pre-trained on Kinetics [57] with $\sigma = 5$. We crop each video feature sequence with overlapped windows of size $L = 256$ and stride 128. In training, we do not use any crops void of actions.

For ActivityNet-1.3 and THUMOS-14, we enumerate all possible combinations of start and end as anchors, *e.g.* $\{(t_s, t_e)|\ 0 < t_s < t_e < L;\ t_s, t_e \in \mathcal{N};\ t_e - t_s < D\}$, where D is 100 for ActivityNet-1.3 and 64 for THUMOS-14. In SGAlign, we use $\tau_1 = 32, \tau_2 = 4$ for ActivityNet-1.3 and $\tau_1 = \tau_2 = 16$ for THUMOS-14.

Training and Inference. We implement and compile our framework using PyTorch 1.1, Python 3.7, and CUDA 10.0. We use $b = 3$ GCNeXt blocks and train our model end-to-end, with batch size of 16. The learning rate is 4×10^{-3} on ActivityNet-1.3 and 6×10^{-6} on THUMOS-14 for the first 5 epochs, and is reduced by 10 for the following 5 epochs. In inference, following [30] to leverage the global video context, we take the video classification scores from [42] and [51], and multiply them by the fused confidence score p_j for evaluation. For post-processing, we apply Soft-NMS [3], where the threshld is 0.84 and select the top-M prediction for final evaluation, where M is 100 for ActivityNet-1.3 and 200 for THUMOS-14. More details can be found in the **supplementary material**.

4.3. Comparison with State-of-the-Art

ActivityNet-1.3: Tab. 1 compares G-TAD with state-of-the-art detectors. We report mAP at different tIoU thresholds, as well as average mAP. G-TAD reports the highest average mAP results on this large-scale and diverse dataset. Notably, G-TAD reaches an mAP of 9.02% at IoU 0.95, indicating that the localization is more accurate than others.

THUMOS-14: Tab. 2 compares the action localization results of G-TAD and various state-of-the-art methods on the THUMOS14 dataset. At IoU 0.7, G-TAD reaches an mAP of 23.4%, obviously higher than the current best 20.8% of TALNet. At IoU 0.5, G-TAD outperforms all methods except TALNet. Besides, when combined with a proposal post-processing method P-GCN [55], G-TAD performs even better, especially at IoUs ≤ 0.5. Now G-TAD reaches 51.6% at IoU 0.5, outperforming all the other methods. In addition, we also report the results of BSN with P-GCN (directly taken from [55]), which are not as good as G-TAD + P-GCN, albeit showing improvement from BSN. This signifies the advantage of G-TAD proposals regardless of post-processing.

Table 1. **Action detection results on validation set of ActivityNet-1.3**, measured by mAP (%) at different tIoU thresholds and the average mAP. G-TAD achieves better performance in average mAP than the other methods, even the latest work of BMN and P-GCN shown in the second-to-last block.

Method	0.5	0.75	0.95	Average
Wang *et al.* [46]	43.65	-	-	-
Singh *et al.* [40]	34.47	-	-	-
SCC [21]	40.00	17.90	4.70	21.70
CDC [36]	45.30	26.00	0.20	23.80
R-C3D [52]	26.80	-	-	-
BSN [30]	46.45	29.96	8.02	30.03
Chao *et al.* [9]	38.23	18.30	1.30	20.22
P-GCN [55]	48.26	33.16	3.27	31.11
BMN [29]	50.07	**34.78**	8.29	33.85
G-TAD (ours)	**50.36**	34.60	**9.02**	**34.09**

Table 2. **Action detection results on testing set of THUMOS-14**, measured by mAP (%) at different tIoU thresholds. G-TAD achieves the best performance for IoU@0.7, and combined with P-GCN, G-TAD significantly outperforms all the other methods.

Method	0.3	0.4	0.5	0.6	0.7
SST [5]	-	-	23.0	-	-
CDC [36]	40.1	29.4	23.3	13.1	7.9
TURN-TAP[15]	44.1	34.9	25.6	-	-
CBR [16]	50.1	41.3	31.0	19.1	9.9
SSN [56]	51.9	41.0	29.8	-	-
BSN [30]	53.5	45.0	36.9	28.4	20.0
TCN [11]	-	33.3	25.6	15.9	9.0
TAL-Net [9]	53.2	48.5	42.8	33.8	20.8
MGG [32]	53.9	46.8	37.4	29.5	21.3
DBG [28]	57.8	49.4	39.8	30.2	21.7
Yeung *et al.* [53]	36.0	26.4	17.1	-	-
Yuan *et al.* [54]	36.5	27.8	17.8	-	-
Hou *et al.* [22]	43.7	-	22.0	-	-
SS-TAD [4]	45.7	-	29.2	-	9.6
BMN [29]	56.0	47.4	38.8	29.7	20.5
G-TAD (ours)	54.5	47.6	40.2	30.8	**23.4**
BSN+P-GCN [55]	63.6	57.8	49.1	-	-
G-TAD+P-GCN	**66.4**	**60.4**	**51.6**	**37.6**	22.9

Table 3. **Ablating GCNeXt Components.** We disable temporal/semantic graph convolutions and set different cardinalities for detection on ActivityNet-1.3.

GCNeXt block			tIoU on Validation Set			
Temp.	Sem.	Card.	0.5	0.75	0.95	Avg.
✗	✗	1	48.12	32.16	6.41	31.65
✓	✓	1	50.20	**34.80**	7.35	33.88
✓	✗	32	50.13	34.17	8.70	33.67
✗	✓	32	49.09	33.32	8.02	32.63
✓	✓	32	**50.36**	34.60	**9.02**	**34.09**

Table 4. **Ablating SGAlign Components.** We disable the sample-rescale process and the feature concatenation from the semnantic graph for detection on ActivityNet-1.3. The rescaling strategy leads to slight improvement, while the main gain arises from the use of context information (semantic graph).

SGAlign		tIoU on Validation Set			
Samp.	Concat.	0.5	0.75	0.95	Avg.
✗	✗	49.84	34.58	8.17	33.78
✓	✗	49.86	**34.60**	**9.56**	33.89
✓	✓	**50.36**	**34.60**	9.02	**34.09**

4.4. Ablation Study

GCNeXt Module: We ablate the three main components of GCNeXt, mainly GCN on temporal edges, GCN on semantic edges, and cardinality increase. Tab. 3 reports the performance on ActivityNet-1.3, where each component is separately enabled/disabled. We see how each of these components contributes to the performance of the final G-TAD model. We highlight the gains from the semantic graph, showing the benefit of integrating adaptive context from semantic neighbors. It also shows cardinality 32 mostly outperforms cardinality 1.

SGAlign Module: The incorporation of semantic features in SGAlign aggregates more semantic context into each sub-graph, which benefits the subsequent localization compared to merely using the GCNeXt features. The sampling interval s in Alg. 1 is adaptively computed for each sub-graph, leading to better performance than a fixed value (*e.g.* $s = 1$). Tab. 4 shows the effect of semantic features and the sampling strategy from both temporal and semantic graphs on ActivityNet-1.3. While sampling densely gives us minor improvements, we obtain a larger gain by including context information from the semantic graph.

Sensitivity to Video Length: We report the results of the sensitivity of G-TAD to different window sizes in THUMOS-14 in Tab. 5. G-TAD benefits more from larger window sizes ($L = 256$ vs. 128). Larger windows mean that G-TAD can aggregate more context information from the semantic graph. Performance degrades at $L = 512$, where GPU memory limits the batch size and network training is influenced.

Table 5. **Effect of Video Size.** We vary the input video size (window length L) and see that G-TAD performance improves with larger sizes ($L = 256$). Degradation occurs at $L = 512$, since GPU memory limits the batch size to be significantly reduced, leading to a noticeable performance drop.

Window	tIoU on Validation				
Length L	0.3	0.4	0.5	0.6	0.7
64	47.07	39.29	32.25	23.88	15.17
128	51.75	44.90	38.70	29.03	21.32
256	**54.50**	**47.61**	**40.16**	**30.83**	**23.42**
512	48.32	41.71	34.38	26.85	19.29

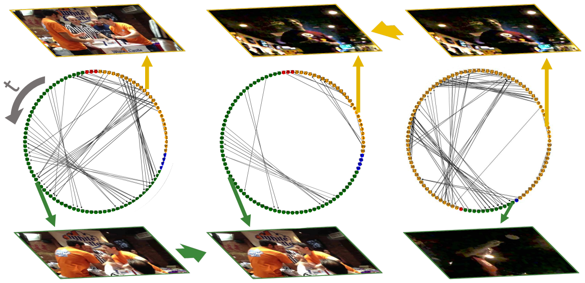

Figure 5. **Semantic graphs and Context**. Given two videos (left and right), we combine action frames of one video with background frames of another to create a synthetic video with no action context (middle). As expected, the semantic graph of the synthetic video contains no edges between action and background snippets.

4.5. Discussion of Action Context

In the ablation study, graph convolutions on the semantic graph improve G-TAD performance in both the GCNeXt block and in the SGAlign layer. Semantic edges connecting background to action snippets can adaptively pass the action context information to each possible action. In this section, we define 2 extra experiments to show how semantic edges encode meaningful context information.

Zero-Context Video. How zero context between action and background leads to semantic graphs with no action-background edges is visually shown by comparing semantic graphs resulting from natural videos and synthetically compiled ones. In Fig. 5 (left and right), we present two natural videos that include actions "wrestling" and "playing darts", respectively. Semantic edges in their resultant graphs do exist, connecting action with background snippets, thus exemplifying the usage of context in the detection process. Then, we compile a synthetic video that stacks action frames from the wrestling video and background frames from the darts video, feed it to G-TAD and again visualize the semantic graph (middle). As expected, the semantic graph does not include any action-background semantic edges.

Correlation to Context Amount. We show the correlation between context edges and context as defined by human annotators. We define the video **context amount** as the av-

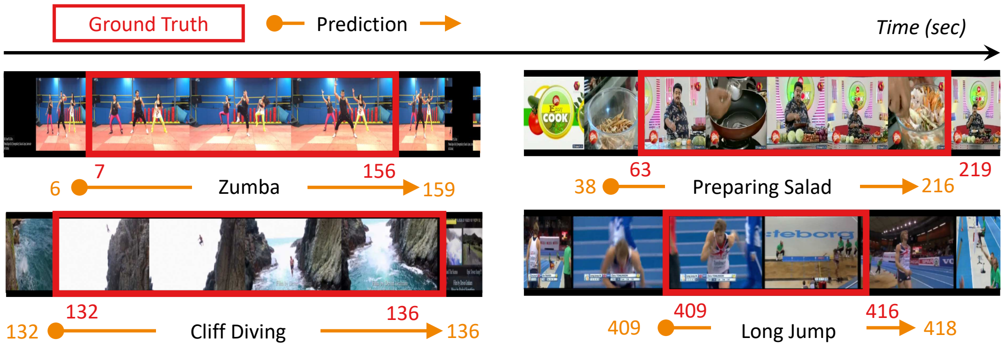

Figure 6. **Qualitative results.** We show qualitative detection results on ActivityNet-1.3 (top) and THUMOS-14 (bottom).

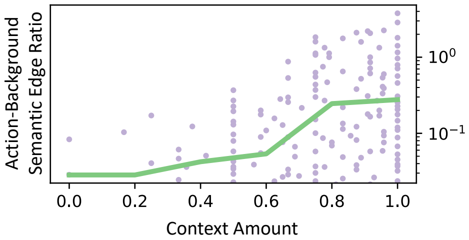

Figure 7. **Action-Background Semantic Edge Ratio vs. Context Amount.** In the scatter plot, each purple dot corresponds to a different video graph. Strong positive correlation is observed between context amount and action-background semantic edge ratio, which means we predict on average more semantic edges in the presence of larger video context.

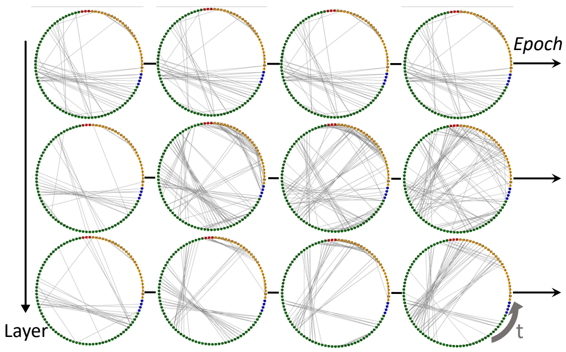

Figure 8. **Semantic graph evolution during G-TAD training.** We visualize the semantic graphs at first, middle, and last layers during training epoch 0, 3, 6, and 9. The semantic edges at the first layer are always the same, while the semantic graphs at the middle and last layers evolve to incorporate more context.

erage number of background snippets which can be used to predict the foreground class. Following DETAD [1], we collect context amount for all videos in ActivityNet validation set from Amazon Mechanical Turk. The scatters in Fig. 7 shows the relation between **context amount** and the ratio of **action-background** semantic edges over all the semantic edges. From the plot, we observe that if a video has a higher amount of context, it is more likely to have more action-background semantic edges in its semantic graph. We further average the ratios in five context amount ranges, and plot them in green. The strong positive correlation between context amount and action-background semantic edge ratio indicates that our G-TAD model can effectively find related context snippets in the semantic graph.

4.6. Visualization

We show a few qualitative detection results in Fig. 6 on both ActivityNet-1.3 and THUMOS-14. In Fig. 8, we visualize the evolution of semantic graphs during the training process across GCNeXt layers. Specifically, we feed a video into G-TAD and visualize the semantic graphs emerging at the first, middle, and last layers at epochs 0, 3, 6, and 9 of training. The semantic graphs at the first layer are the same, since they are built on the same input features. As we progress to different layers and epochs, semantic graphs adaptively update their edges. Interestingly, we observe the presence of more context edges as training advances. This indicates that G-TAD progressively learns to incorporate multiple levels of context in the detection process.

5. Conclusion

In this paper, we cast the temporal action detection task as a sub-graph localization problem by formulating videos as graphs. We take video snippets as graph nodes, snippet-snippet correlations as edges, and apply graph convolution as the basic operation. We propose a new architecture G-TAD to localize sub-graphs. G-TAD includes GCNeXt blocks to aggregate context features from semantically correlated snippets and an SGAlign layer to transform sub-graph features into vector representations. G-TAD can learn enriched multi-level semantic context in an adaptive way using stacked dynamic graph convolutions. Extensive experiments show that G-TAD can find global video context without extra supervision and achieve the state-of-the-art performance on both THUMOS-14 and ActivityNet-1.3.

Acknowledgments. This work was supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research (OSR) under Award No. OSR-CRG2017-3405.

References

[1] Humam Alwassel, Fabian Caba Heilbron, Victor Escorcia, and Bernard Ghanem. Diagnosing error in temporal action detectors. In *European Conference on Computer Vision (ECCV)*, 2018.

[2] Humam Alwassel, Fabian Caba Heilbron, and Bernard Ghanem. Action search: Spotting actions in videos and its application to temporal action localization. In *European Conference on Computer Vision (ECCV)*, 2017.

[3] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S. Davis. Soft-nms – improving object detection with one line of code. In *International Conference on Computer Vision (ICCV)*, 2017.

[4] Shyamal Buch, Victor Escorcia, Bernard Ghanem, Li Fei-Fei, and Juan Carlos Niebles. End-to-end, single-stream temporal action detection in untrimmed videos. In *the British Machine Vision Conference (BMVC)*, 2017.

[5] Shyamal Buch, Victor Escorcia, Chuanqi Shen, Bernard Ghanem, and Juan Carlos Niebles. Sst: Single-stream temporal action proposals. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[6] Fabian Caba Heilbron, Juan Carlos Niebles, and Bernard Ghanem. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016.

[7] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[8] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[9] Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A. Ross, Jia Deng, and Rahul Sukthankar. Rethinking the faster r-cnn architecture for temporal action localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[10] Yunpeng Chen, Marcus Rohrbach, Zhicheng Yan, Yan Shuicheng, Jiashi Feng, and Yannis Kalantidis. Graph-based global reasoning networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[11] Xiyang Dai, Bharat Singh, Guyue Zhang, Larry S. Davis, and Yan Qiu Chen. Temporal context network for activity localization in videos. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2009.

[13] Victor Escorcia, Fabian Caba Heilbron, Juan Carlos Niebles, and Bernard Ghanem. Daps: Deep action proposals for action understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[14] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[15] Jiyang Gao, Zhenheng Yang, Kan Chen, Chen Sun, and Ram Nevatia. Turn tap: Temporal unit regression network for temporal action proposals. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[16] Jiyang Gao, Zhenheng Yang, and Ram Nevatia. Cascaded boundary regression for temporal action detection. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2017.

[17] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, 2015.

[18] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. *arXiv preprint arXiv:1906.02739*, 2019.

[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, 2017.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (ICCV)*, 2016.

[21] Fabian Caba Heilbron, Wayner Barrios, Victor Escorcia, and Bernard Ghanem. Scc: Semantic context cascade for efficient action detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[22] Rui Hou, Rahul Sukthankar, and Mubarak Shah. Real-time temporal action localization in untrimmed videos by subaction discovery. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2017.

[23] YG Jiang, J Liu, A Roshan Zamir, G Toderici, I Laptev, M Shah, and R Sukthankar. Thumos challenge: Action recognition with a large number of classes, 2014.

[24] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[25] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[26] Guohao Li, Matthias Müller, Guocheng Qian, Itzel C. Delgadillo, Abdulellah Abualshour, Ali Thabet, and Bernard Ghanem. Deepgcns: Making gcns go as deep as cnns, 2019.

[27] Guohao Li, Guocheng Qian, Itzel C. Delgadillo, Matthias Müller, Ali Thabet, and Bernard Ghanem. Sgas: Sequential greedy architecture search, 2019.

[28] Chuming Lin, Jian Li, Yabiao Wang, Ying Tai, Donghao Luo, Zhipeng Cui, Chengjie Wang, Jilin Li, Feiyue Huang, and Rongrong Ji. Fast learning of temporal action proposal via dense boundary generator, 2019.

[29] Tianwei Lin, Xiao Liu, Xin Li, Errui Ding, and Shilei Wen. Bmn: Boundary-matching network for temporal action proposal generation. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[30] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang. Bsn: Boundary sensitive network for temporal action proposal generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[31] Xingyu Liu, Joon-Young Lee, and Hailin Jin. Learning video representations from correspondence proposals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[32] Yuan Liu, Lin Ma, Yifeng Zhang, Wei Liu, and Shih-Fu Chang. Multi-granularity generator for temporal action proposal. *Computing Research Repository (CoRR)*, 2018.

[33] Fuchen Long, Ting Yao, Zhaofan Qiu, Xinmei Tian, Jiebo Luo, and Tao Mei. Gaussian temporal awareness networks for action localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[34] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[35] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, 2015.

[36] Zheng Shou, Jonathan Chan, Alireza Zareian, Kazuyuki Miyazawa, and Shih-Fu Chang. Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[37] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[38] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, 2014.

[39] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[40] Gurkirt Singh and Fabio Cuzzolin. Untrimmed video classification for activity detection: submission to activitynet challenge. *arXiv preprint arXiv:1607.01979*, 2016.

[41] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, 2015.

[42] Limin Wang, Yuanjun Xiong, Dahua Lin, and Luc Van Gool. Untrimmednets for weakly supervised action recognition and detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[43] Limin Wang, Yuanjun Xiong, Zhe Wang, and Yu Qiao. Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159*, 2015.

[44] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Val Gool. Temporal segment networks: Towards good practices for deep action recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[45] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *Proceedings of the European Conference on Computer Vision*, 2016.

[46] Ruxin Wang and Dacheng Tao. Uts at activitynet 2016. *ActivityNet Large Scale Activity Recognition Challenge*, 2016.

[47] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[48] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay Sarma, Michael Bronstein, and Justin Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics*, 2018.

[49] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2017.

[50] Zhuyang Xie, Junzhou Chen, and Bo Peng. Point clouds learning with attention-based graph convolution networks. *arXiv preprint arXiv:1905.13445*, 2019.

[51] Yuanjun Xiong, Limin Wang, Zhe Wang, Bowen Zhang, Hang Song, Wei Li, Dahua Lin, Yu Qiao, L. Van Gool, and Xiaoou Tang. Cuhk & ethz & siat submission to activitynet challenge 2016. 2016.

[52] Huijuan Xu, Abir Das, and Kate Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, 2017.

[53] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[54] Zehuan Yuan, Jonathan C. Stroud, Tong Lu, and Jia Deng. Temporal action localization by structured maximal sums. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[55] Runhao Zeng, Wenbing Huang, Mingkui Tan, Yu Rong, Peilin Zhao, Junzhou Huang, and Chuang Gan. Graph convolutional networks for temporal action localization. *arXiv preprint arXiv:1909.03252*, 2019.

[56] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. Temporal action detection with structured segment networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[57] Andrew Zisserman, Joao Carreira, Karen Simonyan, Will Kay, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.